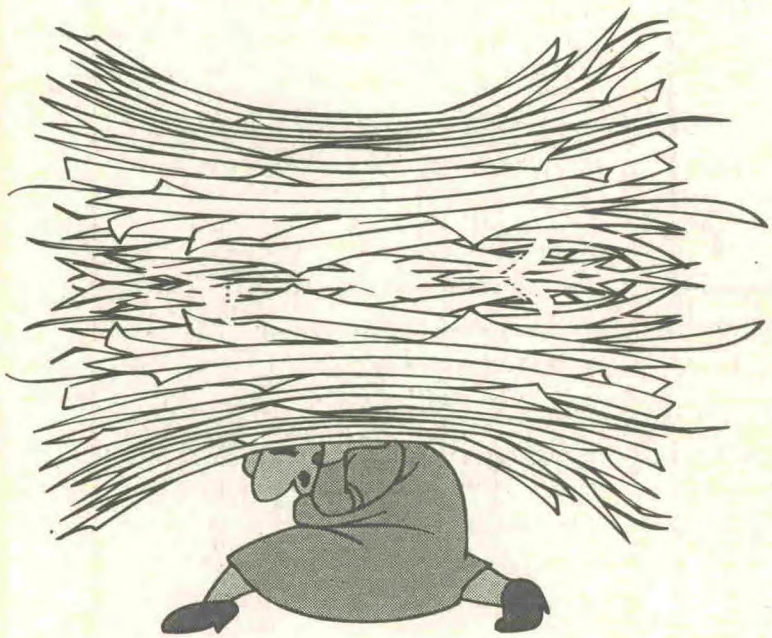
ซอฟต์แวร์

ธาวินทร์ ลีลาคุณารักษ์

สับเพเหระกับเรื่องของ Visual dBASE



ศึกษาเรื่องราวที่น่าสนใจ
เกี่ยวกับ Visual dBASE
ซึ่งไม่มีใน dBASE เดิม

สำหรับคนที่เคยสร้างโปรแกรมประยุกต์ด้วย dBASE III+Clipper หรือ Foxpro ในยุคก่อน ๆ ที่ทำงานบนระบบดอส พอถึงยุคที่ต้องพัฒนางานภายใต้ระบบวินโดวส์อย่างในปัจจุบัน คงจะมีความคิด ความรู้สึกในเรื่องของการโปรแกรมมิ่งที่คล้าย ๆ กัน คือ ส่วนใหญ่เราจะไม่ต้องการให้วิธีการทางด้านโปรแกรมมิ่งของภาษาคอมพิวเตอร์ที่เราถนัดเปลี่ยนแปลงไปจากเดิมมากนัก

ยกตัวอย่างที่เห็นได้ชัดตัวอย่างหนึ่งคือ เรื่องของโปรแกรม Visual Objects หรือ VO ในช่วงก่อนที่โปรแกรมชุดนี้จะออกสู่ตลาด คนที่เคยใช้ Clipper ในการสร้างโปรแกรมประยุกต์ของตน ต่างคาดหวังว่าเราจะสามารถใช้วิธีการเขียนโปรแกรมตามแบบของ Clipper เดิมกับโปรแกรม Visual Objects ได้ แต่ทุกคนก็ต้องผิดหวัง เพราะผู้ผลิตได้เปลี่ยนวิธีการทางโปรแกรมมิ่งของ VO เป็นแบบใหม่หมด จนผู้ใช้ Clipper รับไม่ได้ และถึงตอนนี้แทบจะไม่มีใครหันกลับไปใช้ผลิตภัณฑ์ชุดนี้อีกเลย

สำหรับโปรแกรม dBASE ซึ่งเป็นต้นตำรับของภาษา xBASE เอง ถึงแม้จะมีการพัฒนาเปลี่ยนแปลงอย่างต่อเนื่อง จนกระทั่งเป็น Visual dBASE อย่างในปัจจุบัน ซึ่งโปรแกรมคอมพิวเตอร์ชุดนี้จัดเป็นโปรแกรมจัดการฐานข้อมูลและคอมพายเลอร์ประเภท Visual Programming Concept ที่น่าสนใจอีกโปรแกรมหนึ่ง ถึงแม้ dBASE จะมีการพัฒนาไปมากก็ตาม แต่ในส่วนที่ต้องมีการเขียนชุดคำสั่งเพื่อควบคุมกลไกของงานที่เราสร้างขึ้นแล้ว dBASE ยังคงรูปแบบของการเขียนชุดคำสั่งแบบเดิม ๆ ไว้เป็นส่วนใหญ่ แต่ถึงอย่างไรก็ดี Visual dBASE ในวันนี้ได้มีการกำหนดแนวคิด (concept) ใหม่ ๆ ขึ้นมา หลายอย่างที่ไม่มีใน dBASE รุ่นก่อน ๆ ที่เราน่าจะทำความรู้จักหรือสร้างความคุ้นเคยก่อนที่จะนำโปรแกรมชุดนี้ไปใช้สร้างงานต่าง ๆ โดยในครั้งนี้ผมมีเรื่องที่น่าสนใจเกี่ยวกับ Visual dBASE หลายเรื่องด้วยกันตามลำดับดังต่อไปนี้ครับ

เรื่องของ Classes

ถ้าเราใช้ภาษาโปรแกรมคอมพิวเตอร์ที่เป็น Object Orianted Programming ก็คงจะหนีไม่พ้นที่จะต้องทำความเข้าใจในเรื่องของ class ของภาษานั้น ๆ ว่าเขาเน้นเรื่องของ class ไปในลักษณะใด

สำหรับใน Visual dBASE เรื่องของ class จะหมายถึงการกำหนดหรือการระบุรายละเอียด และการกำหนดกลไกการทำงาน ชิ้นส่วนของโปรแกรมที่ใช้เป็นแม่แบบที่เราจะมาใช้สร้างชิ้นส่วนของโปรแกรม (Object หรือ Component) เพื่อนำมาประกอบกันเป็นโปรแกรมประยุกต์สำหรับงานหนึ่ง ๆ โดย class ใน Visual dBASE จะหมายถึงชิ้นส่วนของโปรแกรมที่เป็นส่วน User Interface เป็นส่วนใหญ่ เช่นเรากำหนดลักษณะของปุ่มคำสั่งแบบใหม่ขึ้นมาใช้การสร้างแม่แบบของปุ่มคำสั่งลักษณะใหม่นี้ก็คือการสร้าง class ขึ้นมาใหม่ชุดหนึ่งนั่นเองครับ

ใน Visual dBASE จะมี class อยู่ 2 ลักษณะ คือ Standard classes เป็น class ต้นแบบของชิ้นส่วนของโปรแกรมที่เป็นมาตรฐานที่ Visual dBASE จัดเตรียมไว้ให้ ซึ่งมีอยู่ด้วยกัน 25 class ดังรายละเอียดในรูปที่ 1 ส่วน class อีกลักษณะหนึ่งคือ Customer classes จะเป็น class ที่เรากำหนดขึ้นมาใช้เอง โดยจะใช้คำสั่ง CLASS.....ENDCLASS ในการสร้าง ซึ่งมีรูปแบบของคำสั่งดังนี้

```
CLASS <ชื่อ class ที่เรากำหนดใหม่>[(<parameter>)]
[OF <ชื่อ class ที่ใช้เป็นต้นแบบสำหรับสร้าง class ใหม่นี้>
[(<parameter>)]]
[CUSTOM]
[FROM <แฟ้มที่บรรจุ code ของ class ต้นแบบ>]
[PROTECT <รายการคุณสมบัติที่ใช้เฉพาะใน class ใหม่>]
<constructor code>
<member function>
ENDCLASS
```

class ใหม่ที่ได้นี้จะสามารถใช้เป็นต้นแบบของชิ้นส่วนของโปรแกรมได้เช่นเดียวกับ class มาตรฐาน และในการนำ class หรือต้นแบบของชิ้นส่วนของโปรแกรมไปสร้างเป็น Object ใหม่ หรือชิ้นส่วนของโปรแกรมที่เราจะนำไปใช้นั้น จะใช้คำสั่ง DEFINE ในการสร้าง Object ใหม่จาก class ที่มีอยู่เดิม รูปแบบของคำสั่ง DEFINE มีดังนี้

```
DEFINE <ชื่อ class มาตรฐานหรือต้นแบบ> <ชิ้นส่วนหรือ Object ที่สร้างขึ้นใหม่>
[OF <ชื่อชิ้นส่วนที่ Object นี้บรรจุอยู่>]
[FROM <row,col> TO <row,col> | AT <row,col>]
[PROPERTY <รายการของคุณสมบัติของชิ้นส่วนใหม่โดยใช้รายการจากต้นแบบ>]
[CUSTOM <รายการของคุณสมบัติที่กำหนดขึ้นมาใหม่>]
```

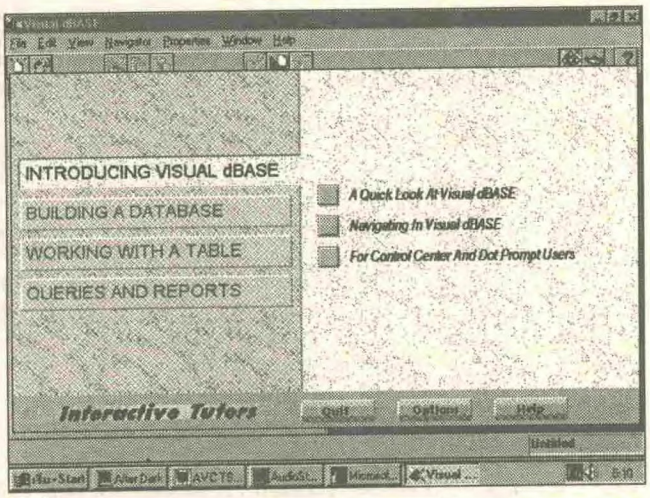
Browse	
Checkbox	
Compobox	
DDELink	สร้าง Object ที่ใช้สื่อสารโยงกับ DDE Server
DDETopic	สร้าง Object ที่ใช้เป็น DDE Server เพื่อใช้สื่อสารโยงจากโปรแกรมอื่น ๆ
Editor	
Entryfield	
Form	ตัวฟอร์มที่ใช้เป็นหน้าจอของโปรแกรม
Image	
Line	
Listbox	
Menu	สร้างเมนูตามมาตรฐานของวินโดวส์
Menubar	สร้างส่วนเมนูบาร์ตามมาตรฐานของวินโดวส์
Object	สร้าง Object ชนิดใหม่
Ole	
Paintbox	
Popup	สร้างส่วนชี้ลงมาของเมนูตามมาตรฐานของวินโดวส์
Pushbutton	
Radiobutton	
Rectangle	
Scrollbar	
Shape	
Spinbox	
Tabbox	
Text	

รูปที่ 1
class มาตรฐาน
ใน Visual dBASE

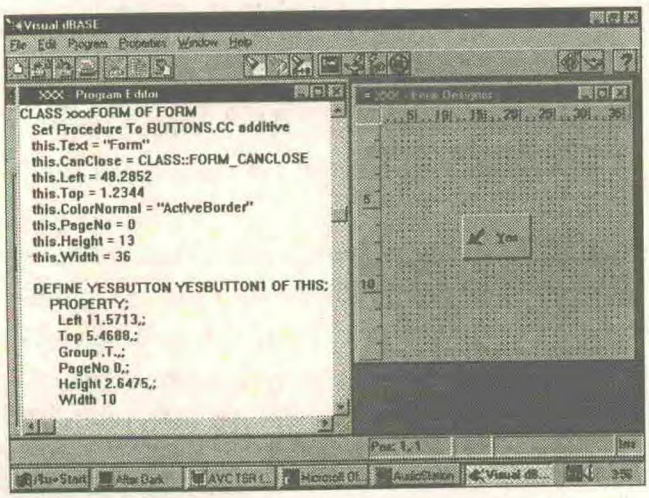
```
-Listing #1 xxx.wfm
-
-**    END HEADER - do not remove this line*
-*     Generated on 27/02/1997
-*
-parameter bModal
-local f
-f = new xxxFORM()
-if (bModal)
-     f.mdi = .F. && ensure not MDI
-     f.ReadModal()
-else
-     f.Open()
-endif
-CLASS xxxFORM OF FORM
-     Set Procedure To BUTTONS.CC additive
-     this.Text = "Form"
-     this.CanClose = CLASS::FORM_CANCLOSE
-     this.Left = 48.2852
-     this.Top = 1.2344
-     this.ColorNormal = "ActiveBorder"
-     this.PageNo = 0
-     this.Height = 13
-     this.Width = 36
-
-   DEFINE YESBUTTON YESBUTTON1 OF THIS;
-       PROPERTY;
-     Left 11.5713,;
-     Top 5.4688,;
-     Group .T.,;
-     PageNo 0,;
-     Height 2.6475,;
-     Width 10
-
-     Procedure Form_CanClose
-
-ENDCLASS
```

ในลิสติงที่ 1 เป็นซอร์สโค้ดของฟอร์มตัวอย่างชื่อ xxx.wfm โดยในโปรแกรมชุดนี้จะประกอบด้วยชิ้นส่วนโปรแกรม (component) 2 ชุด คือ ตัวฟอร์ม ในที่นี้จะสร้างเป็น class ใหม่ชื่อ xxxFORM ส่วนชิ้นส่วนของโปรแกรมอีกชุดหนึ่งคือปุ่มคำสั่ง "Yes" ซึ่งใช้คำสั่ง DEFINE ในการสร้างโดยกำหนดชื่อเป็น YESBUTTON1 ในตัวอย่างนี้เป็นตัวอย่างสั้น ๆ เพื่อให้เห็นวิธีการเขียนโปรแกรมสำหรับสร้าง class และกำหนดชิ้นส่วนของโปรแกรมอื่น ๆ ด้วยคำสั่ง DEFINE

สำหรับในทางปฏิบัติเราคงไม่ต้องกังวลกับวิธีการเขียนชุดคำสั่งเพื่อสร้าง class หรือกำหนดชิ้นส่วนต่าง ๆ ของโปรแกรมที่เราจัดทำขึ้น เพราะปกติเราจะสร้างส่วนที่เป็น User Interface หรือหน้าจอของโปรแกรมใน Visual Mode อยู่แล้ว ซึ่งเมื่อเราสร้างงานส่วนนี้เสร็จ โปรแกรม Visual dBASE จะทำการเจนเนอร์เรจเป็นซอร์สโค้ดภาษา dBASE ให้ทันที ดังตัวอย่างในรูปที่ 3 แต่เหตุที่ผมหยิบยกขึ้นมาคุย ก็เพื่อให้เกิดความคุ้นเคยกับลักษณะของการโปรแกรมด้วยภาษา dBASE ในยุคนี้ครับ



รูปที่ 2 Interactive Tutors ของ Visual dBASE



รูปที่ 3 class ที่ Visual dBASE สร้างตอนเราสร้างหน้าจอของโปรแกรมประยุกต์

ข้อมูลชนิดใหม่ใน Visual dBASE

สำหรับชนิดของข้อมูล (Data Type) ที่ถูกกำหนดขึ้นมาใหม่ในรุ่นของ Visual dBASE นี้ มีอยู่หลายชนิดด้วยกัน บางชนิดใช้กำหนดในฟิลด์เก็บข้อมูลในตารางข้อมูลแบบ dBASE ได้ แต่บางชนิดจะใช้เฉพาะตอนสร้างชุดคำสั่งเท่านั้น ชนิดของข้อมูลที่เกิดขึ้นใหม่ที่สนใจมีดังนี้ครับ

```
-Binary (B) และ Ole (G)
```

เป็นชนิดของข้อมูลที่ใช้เก็บข้อมูลไบนารีมาตรฐาน (Binary) และเก็บข้อมูลที่มีการเชื่อมโยงแบบ OLE ซึ่งจะใช้เก็บข้อมูลจำพวกภาพเคลื่อนไหว เสียง รูปภาพ หรือข้อมูลลักษณะพิเศษของระบบวินโดวส์ เราสามารถกำหนดชนิดข้อมูลทั้งแบบ Binary หรือ OLE ในฟิลด์ของตารางข้อมูลของ dBASE ได้ด้วย

```
-Bookmark (BM)
```

เป็นชนิดของข้อมูลที่ใช้ระบุตำแหน่งของรายการ (record No.) ในตารางข้อมูลที่ไม่ใช่ตารางข้อมูลแบบ dBASE เช่นตารางข้อมูลแบบ Paradox หรือ SQL เนื่องจากตารางข้อมูลเหล่านี้มักจะไม่ได้ระบุตำแหน่งรายการเป็นลำดับเลขตามรายการที่ 1,2,3,... แต่จะมีลำดับรายการเป็นอย่างอื่น ข้อมูลชนิด Bookmark นี้จะเกิดขึ้นเมื่อมีการเรียกใช้คำสั่ง BOOKMARK() เช่น

```
USE B_name.db
GO TOP
mark1  = BOOKMARK()
...
```

ตัวแปร mark1 จะเป็นตัวแปรแบบ Bookmark ซึ่งเก็บตำแหน่งรายการข้อมูลจากตารางข้อมูลที่ส่งออกมาจากคำสั่ง BOOKMARK() ข้อมูลแบบ Bookmark นี้เราจะเรียกมาดูไม่ได้ (non-printing value) แต่เราสามารถใช้ในการเปรียบเทียบทางตรรกะได้

```
-NULL (U)
```

ความจริง NULL เป็นค่าของข้อมูลมากกว่าที่จะเป็นชนิดของข้อมูลคือเป็นค่าที่จะเก็บในตัวแปรหรือฟิลด์ในตารางข้อมูล แต่จะไม่ใช้ในการกำหนดหรือสร้างตัวแปรใหม่หรือไม่ได้ใช้ในการกำหนดเป็นชนิดของข้อมูลในฟิลด์ข้อมูลเหมือนชนิดของตัวแปรแบบอื่น ๆ

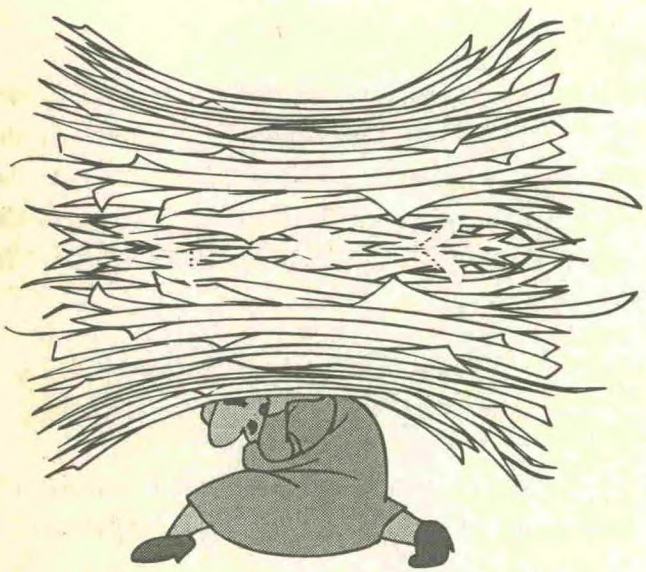
ซึ่งลักษณะพิเศษของ NULL ก็คือ ค่าใด ๆ ก็ตามที่มากระทำกับ NULL (Operate) จะให้ค่าเป็น NULL ตามลักษณะชนิดของข้อมูลของค่านั้นเสมอ สำหรับค่า NULL จะใช้กับตารางข้อมูลแบบ dBASE ไม่ได้ แต่จะใช้กับตารางข้อมูลแบบอื่น ๆ ที่สนับสนุนการใช้ค่า NULL เท่านั้น

```
-Function-Pointer (FP)
```

ถ้าเราใช้หรือศึกษาการใช้โปรแกรม Visual dBASE ในช่วงแรก ๆ เราจะเห็นวิธีการเขียนชุดคำสั่งแบบแปลก ๆ ที่เราไม่เคยใช้ใน Clipper หรือ dBASE รุ่นก่อนเช่น

```
DEFINE PUSHBUTTON  Next OF dataForm
      PROPERTY
      OnClick    go_next
      ....
      ....
PROCEDURE go_next
      IF .NOT. EOF()
      SKIP
      ELSE
      GO TOP
      ENDIF
```

คุณสมบัติของ Object (Property) ที่ชื่อ OnClick นี้จะเป็นตัวแปรแบบ Function -Pointer ซึ่งจะใช้เชื่อมโยงกับ PROCEDURE หรือ FUNCTION ที่เราต้องการให้คำสั่งเหล่านั้นทำงาน เมื่อถูกการกระตุ้นที่ตรงกับเงื่อนไขของกลไกของ Object ที่เรากำหนดไว้ ข้อมูล



ชนิด Function -Pointer นี้จะใช้กับการสร้างชุดคำสั่งเท่านั้น ไม่ได้ใช้กำหนดเป็นชนิดของฟิลด์ข้อมูลในตารางข้อมูล

```
-Codeblock (CB)
```

เราเคยได้ยินเรื่องของ codeblock มาบ้างแล้วนะครับ โดยเฉพาะคนที่เคยใช้ Clipper คงจะรู้จักดีใน Visual dBASE ก็มีให้ใช้เหมือนกันครับ และใช้วิธีการเดียวกับของ Clipper ด้วย ตัวแปรแบบ codeblock จะมีลักษณะเหมือนกับชื่อของ PROCEDURE หรือ FUNCTION ที่เรากำหนดขึ้น codeblock ใน Visual dBASE จะมี 2 ลักษณะ คือ

1. Expression Codeblock ใช้สร้างชุดคำสั่งที่เกี่ยวกับการคำนวณต่าง ๆ รูปแบบการสร้างชุดคำสั่งมีดังนี้

```
-{[|<parameter>|] <expression>}-
```

ตัวอย่างเช่น Mult = { |a,b| a*b }

ตัวแปร Mult จะเป็นตัวแปรแบบ codeblock ซึ่งภายใน codeblock จะเก็บชุดคำสั่ง a*b ไว้ โดยมีการผ่านค่าทาง |a,b| เวลาเรียกใช้จะได้ชุดคำสั่งดังนี้

```
x = Mult(5,10) && x มีค่าเท่ากับ 50
```

2. Statement Codeblock ใช้สร้างกลุ่มคำสั่งเพื่อทำงานใดงานหนึ่งด้วยชุดคำสั่งของ dBASE รูปแบบของ statement codeblock มีดังนี้

```
-{[|parameter|];<statement>[;statement]}-
```

ตัวอย่างเช่น Mult = {|a,b|;? a*b}

คำสั่งนี้จะต่างจากตัวอย่างก่อนตรงที่คำสั่งนี้เราต้องการใช้โปรแกรมพิมพ์ค่าผลคูณออกมาแทนที่จะส่งค่าไปเก็บที่ตัวแปร เวลาเรียกใช้จะได้ชุดตัวอย่างชุดคำสั่งดังนี้

```
Mult(5,10) &&โปรแกรมจะพิมพ์เลข 50 ทาง device ที่กำหนด
```

```
-Object-Reference (O)
```

ใน Visual dBASE มีวิธีการสร้าง Object ขึ้นมาใหม่อีกวิธีหนึ่ง โดยใช้ NEW Operator ในการสร้างตัวอย่างเช่น

```
mainForm = NEW Form()
```

ตัว Object ที่ชื่อ mainForm ที่สร้างขึ้นมาใหม่นี้จะไม่ใช่ฟอร์ม (FORM) จริง ๆ แต่เป็นตัวแปรที่อ้างอิงกับ FORM เท่านั้น ซึ่งตัวแปรลักษณะนี้เราจะเรียกว่าตัวแปรแบบ Object Reference โดยตัวแปรที่ถูกสร้างด้วยวิธีนี้จะมีรายละเอียดเหมือนกับ Object จริงทุกประการ ดังนั้นเวลาเราจะดำเนินการใด ๆ กับตัวแปรแบบ Object Reference นี้ก็จะดำเนินการเหมือนที่เราดำเนินการกับ Object จริง เช่น จากตัวอย่างข้างต้น ถ้าเราต้องการเปลี่ยนชื่อที่แสดงในส่วนไตเติลของ mainForm ใหม่เป็น "MAIN" เราจะเขียนชุดคำสั่งดังนี้

```
mainForm.Text = "MAIN"
```

ในตัวอย่างนี้เราจะใช้วิธีการที่เรียกว่า dot operator เพื่อเรียกคุณสมบัติหัวข้อ "Text" ซึ่งเป็นหัวข้อที่เก็บชื่อไตเติลของฟอร์มขึ้นมาแก้ไข ในทางปฏิบัติเราจะมองตัวแปรแบบ Object Reference นี้เป็นเสมือน Object ชุดหนึ่ง ครับเรื่องราวของ Visual dBASE ที่ผมนำมาเสนอในครั้งนี้ ส่วนใหญ่จะเป็นเรื่องใหม่ที่ไม่มีใน dBASE เดิม หรือเป็นเรื่องที่ไม่คุ้นเคยกันนัก ความจริงแล้วใน Visual dBASE ยังมีอีกหลาย ๆ เรื่องที่จำเป็นต้องนำมาศึกษาเพื่อให้เกิดความคุ้นเคยก่อน โดยเฉพาะที่เกี่ยวกับการสร้างชุดคำสั่งเพื่อควบคุมกลไกของส่วนต่าง ๆ ของโปรแกรมประยุกต์ที่เราจัดทำขึ้น คงต้องขอยกยอดไปคราวหน้านะครับ ถ้าผมมีเวลาว่างจะพยายามนำตัวอย่างง่าย ๆ เพื่อใช้สำหรับศึกษาเรื่องของ Visual dBASE ควบคู่กันไปด้วย พบกันใหม่คราวหน้าครับ